Examiner's Amendment and Statement of Reasons for Allowance

1. This action is responsive to Applicant's amendment filed October 09, 2008.

2. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on October 09, 2008.has been entered.

Examiner's Amendment

3. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Mr. Sean F. Sullivan, Registration Number 38, 328, on December 01, 2008 for obviating any potential 101 issues and put the claims in condition for allowance.

The application has been amended as follows:

1. (currently amended) A method of creating a dynamically linked computer program function package, the method comprising:

establishing an attribute, each attribute exhibiting a plurality of at least one of variations, characteristics and parameters associated with a specific computer program function;

obtaining a source file associated with said specific computer program function;

compiling said source file iteratively to create a plurality of corresponding object files based on said at least one of variations, characteristics, and parameters for each said attribute, wherein each iteration of the compiling said source file results in the plurality of object files having the specific computer program function but with each object file having distinct attributes from one another; and

linking the plurality of resulting object files to create a single executable file, such that the single executable file comprises different versions of the specific computer program function; and

wherein said single executable file is configured to facilitate choice of a selected version of said specific computer function based on a particular said at least one of variations, characteristics, and parameters for each said attribute; and

wherein said attribute includes: version, addressability, character code base, character set that a specific operating system supports, system linkage conventions; machine architecture, or floating point hardware; and

wherein said at least one of variations, characteristics, and parameters for each said attribute includes, 64-bit versus 32-bit addressing; ASCII versus

EBCDIC, internal representation for character data, or HEX versus IEEE representation to use for floating point data.

4. (cancelled)

7. (cancelled)

-- The End --

Examiner's Statement of Reason(s) for Allowance

4. Claims 1-3, 5, 6, and 8 are allowed.

5. The following is an examiner's statement of reasons for allowance:
The prior arts of record: **Jennings** et al., teaches a first dynamic link library (DLL) of a first computing environment, which exports one or more procedures that an application program executing in the first computing environment can call, is replaced with a second DLL that executes in a second computing environment, in a manner that is transparent to the calling application. A source code skeleton of the second DLL is automatically generated based on information contained in a directory of the compiled object code of the first DLL. The calling application executing in the first computing environment is the linked dynamically to the exported procedures of the second DLL in the second computing environment in a manner that is transparent to the calling application. **Srivastava**, teaches Techniques used in a relational database system for defining subclasses of built-in classes and thereby achieving columns in database tables which contain polymorphic objects of the built-in classes. The methods for the subclasses are contained in named packages. A table in the database system's schema relates the package name to a storage location managed by the relational database system that contains the compiled code for the package. When a method from the package is invoked, a dynamic linker uses the name and the table to locate the compiled code and then executes the compiled code. Users may define their own subclasses by writing a package and then compiling the package using a compiler belonging to the database system. The compiler puts the compiled

code in one of the storage locations and modifies the table so that the package name is related to the storage location. **Chen** et al., teaches a framework for a query compiler and run-time environment for resolving a table reference to a dynamic table that is first identified at run-time but is initially unknown at compile-time. A parser parses the table reference and creates a parsed representation for the table that identifies the type of dynamic table. A code generator creates executable plans containing run-time table object representations (TAOB), from the parsed representations, that contain the type of dynamic table. **Mitchell** et al., teaches a method and system for creating named relations between classes in a dynamic object-oriented programming environment via mappers is disclosed. The mapping objects dynamically bind to the class interfaces of the classes being related. These connections between classes are defined within a visual environment. The relationships can be programmatically attached by name to object instances during program execution. **Lawrence** et al., teaches a human oriented object programming system provides an interactive and dynamic process for the incremental building of computer programs which facilitates the development of complex computer programs such as operating systems and large applications with graphic user interfaces (GUIs). The program is modeled as a collection of units called components. A component represents a single compilable language element such as a class or a function. The major functionalities are the database, the compiler, build and link mechanism. The database stores the components and properties. The compiler, along with compiling the source code of a property, and generating object code is responsible for calculating the dependencies associated with a component. **Hunt**, teaches library links to a compiled application using the following variation of

load-time dynamic linking. At some point prior to linking, a user selects a library for linking to the compiled application. An association is made between the selected library and any external libraries referenced within the compiled application. For example, if the application is in Common Object File format, a new import table lists the selected library and the external libraries of the original import table. At link time, the selected library and the external libraries link to the compiled application. At load time, the application, selected library, and any external libraries load. When the selected library loads first, a function in the selected library performs operations before the application or external libraries load. And **Blandy**, teaches a method and apparatus for just in time compilation of Java bytecode methods and provides a means to compile only the paths that are actually executed. An iterative process is employed whereby bytecodes are compiled up to the next conditional flow bytecode or return, the compiled code is executed and any attempt to enter uncompiled paths of the method is monitored. When the executing thread attempts to execute an uncompiled path control is returned to the compiler and the process is repeated starting with the first bytecode of that path. The process continues until all paths have been compiled or until it is deemed likely that all the most commonly executed paths have been compiled. New arts made of record: US Patent No. 7,299,462 by **Shann** et al., teaches a method of preparing an executable program from a plurality of object code modules, at least one of said object code modules including section data specifying a plurality of functions associated with relocation instructions, at least some of which functions are called in the executable program. The method comprises the steps of assigning an attribute to each function, said attribute being capable of providing an indication of whether the function is reachable, reading

the section data and relocation instructions to ascertain if the function is called and setting the attribute to indicate the called status and preparing the executable program to only include functions with an indicated called status of reachable. A linker is provided for preparing the executable program from object code modules containing the relocation instructions. US Patent No. 5,923,882, by **Ho** et al., teaches a system and method of eliminating some of the indirect addressing associated with Position Independent Code (PIC). The invention applies cross-module optimization to dynamic linking of shared libraries. A definition table is built that defines where each symbol within an application program and associated shared library is defined. Also stored in the definitions table is an associated attribute. And US Patent No. 6,334,213, by **Li**, teaches a method for modifying an original executable by injecting it with an injection executable, using a code injection utility. The original executable and the injection executable are of the same or compatible file formats, and the structure of both executables is either known in advance or ascertainable, i.e., by "dumping" and analyzing the contents of the executable. Preferably, the injection executable is a complete, self-contained executable written using standard development tools, such as a graphical, object-oriented development environment. However, none of them, taken alone or in combination, teaches the features in such a manner as recited in independent claim 1.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 8:00am - 4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Chih-Ching Chow/
Examiner, Art Unit 2191
12/01/2008

/Wei Y Zhen/
Supervisory Patent Examiner, Art Unit 2191